ESE507 Project3 Report

Questions

1.
The module in the systermverilog code was designed to use parameters for M and N so that it is able to handle different sizes of matrices and vectors for the system. Since the module uses M and N as parameters, the only change that would need to be made is in the top level module that instantiates the control logic, the control module is instantiated with the required M and N dimensions of the matrices. Furthermore, the control logic also gained readability this time. From our last project, we intentionally used variables to create delays, indeed, we replaced them with counters to simplify the code and logic. For M changes, the ROM size will increase, as the N changes, both ROM and the vector register-based memory will increase the size. Lastly, as the parameter T grows, those registers that accept the data will also increase their bit size.

2.
In C++, there is a conditional to check whether or not P > 1 and if P is 1. If P == 1 then it will be code resembling Part 1. When P > 1, it will look similar to Part 1 however it will now have multiple datapath modules for as many P there is (ie. if P = 2, then there will be 2 instantiated datapath modules). The same is done for ROM when P > 1 as 2 (or more) datapath modules will be computing concurrently different calculations, therefore it is unnecessary for both ROMs to contain the entire matrix, it would be more storage efficient to split the matrix evenly between the 2 (or more) ROMs, 1 ROM for each datapath, to save on costs. For the system verilog side the top level module will need to instantiate the MAC unit P times. The control module is adapted to the parallelism with parameter P passing so it knows when the whole computation or each row computation is finished. When P > 1, a demux with a counter is also needed in the top level module to switch outputting different values generated from each MAC in order. For this design we can still use the pipelined multiplier to increase the clock frequency. On the other hand, if concerned about the area and power cost, then the pipeline register can be omitted to just 1 instead of P of it. However, this will also create more overhead(clock cycles to process product values).

3.

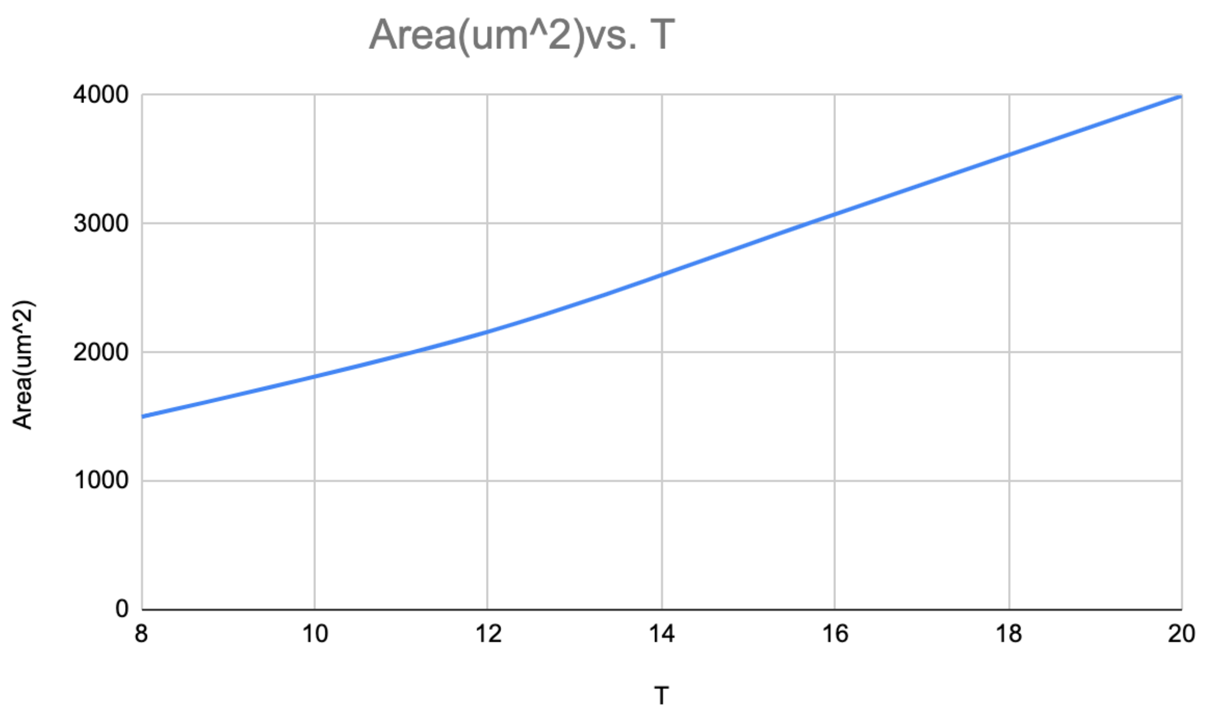

Figure 3.1: This is a plot showing the change in area as T increases.

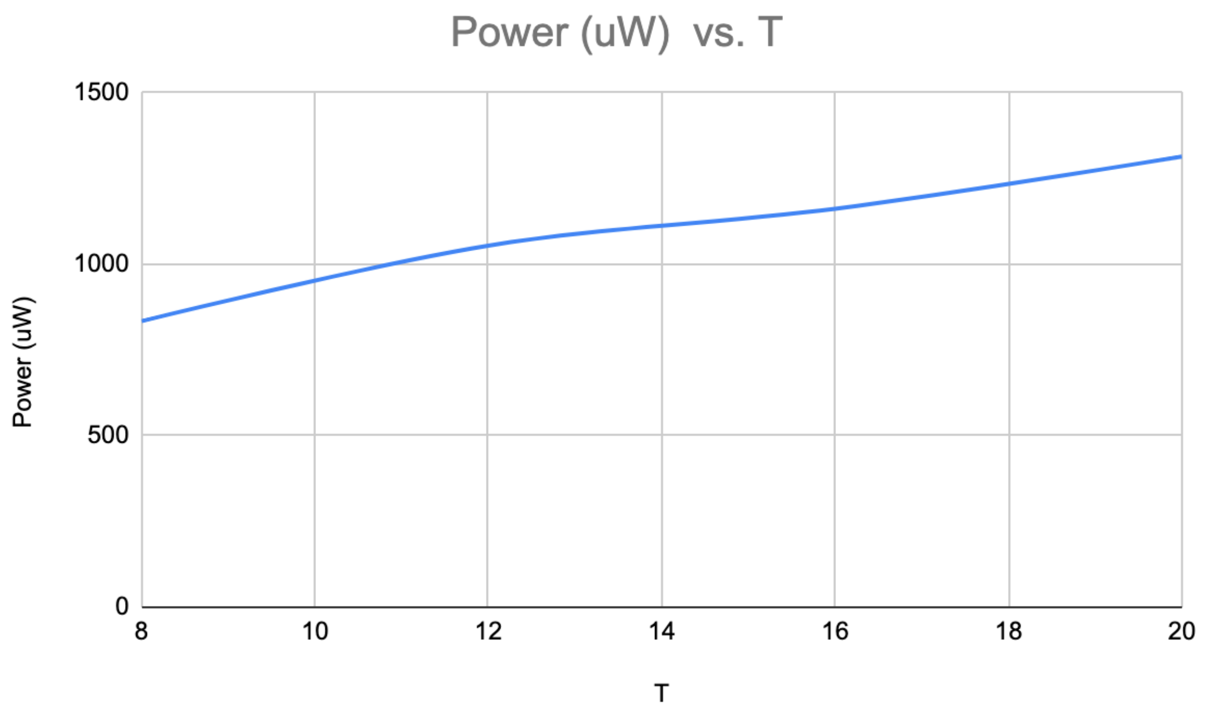

Figure 3.2: This is a plot showing the change in power in uW as T increases.

The critical path for fc_6_6_8_1_1 is from the ROM output to the output of the pipeline register. and the fc_6_6_12_1_1 is looping through the accumulator and adder. In fc_6_6_16_1_1 the critical path changes to be from the vector memory to the output of the pipeline register. In fc_6_6_20_1_1 the critical path is still from the vector memory to the output of the pipeline register. This shows that, increasing the data bit will slightly change the critical path. Also from above plots, it also increases the power and the area as well since the allocation for each data now will need to be extended.

4.

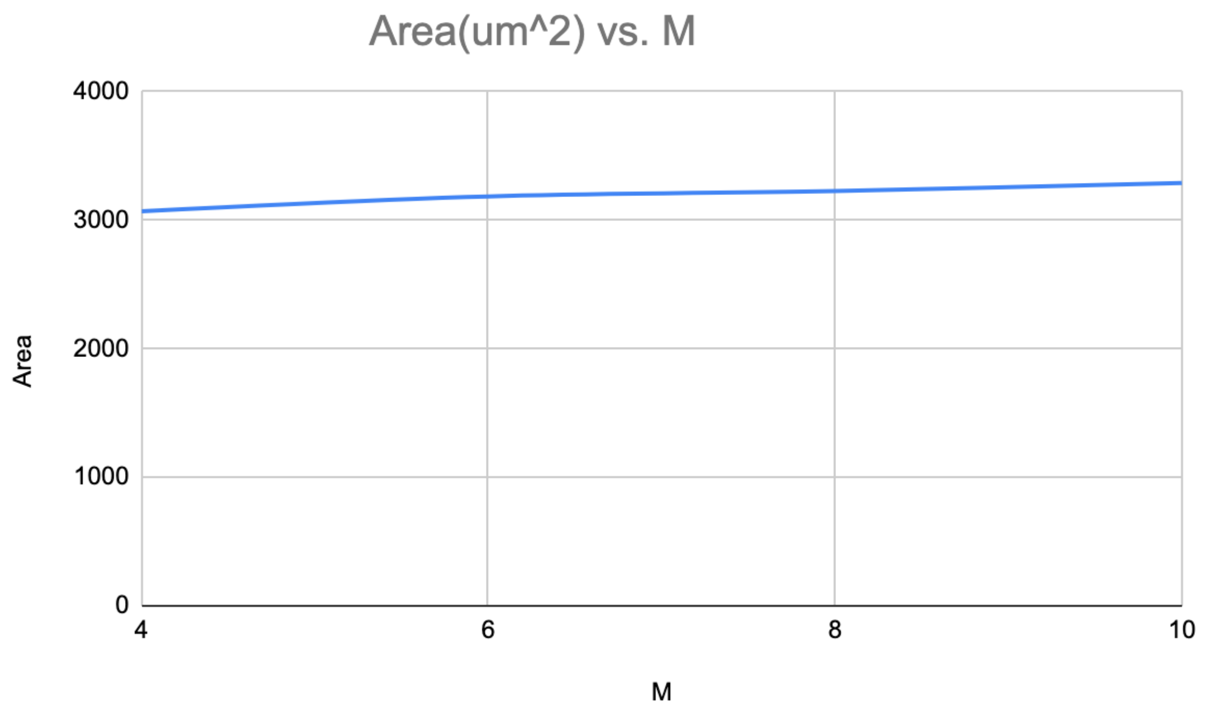

Figure 4.1: This plot shows the change in area as M changes.

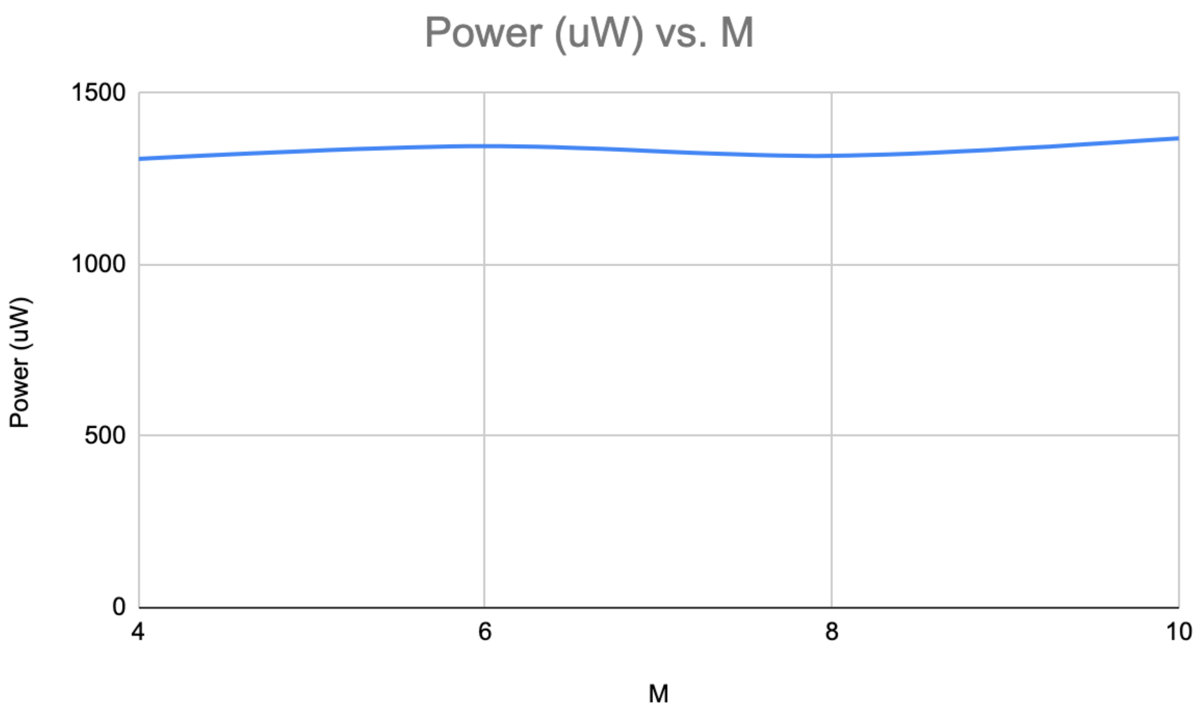

Figure 4.2: This plot shows the change in power in uW as M changes.

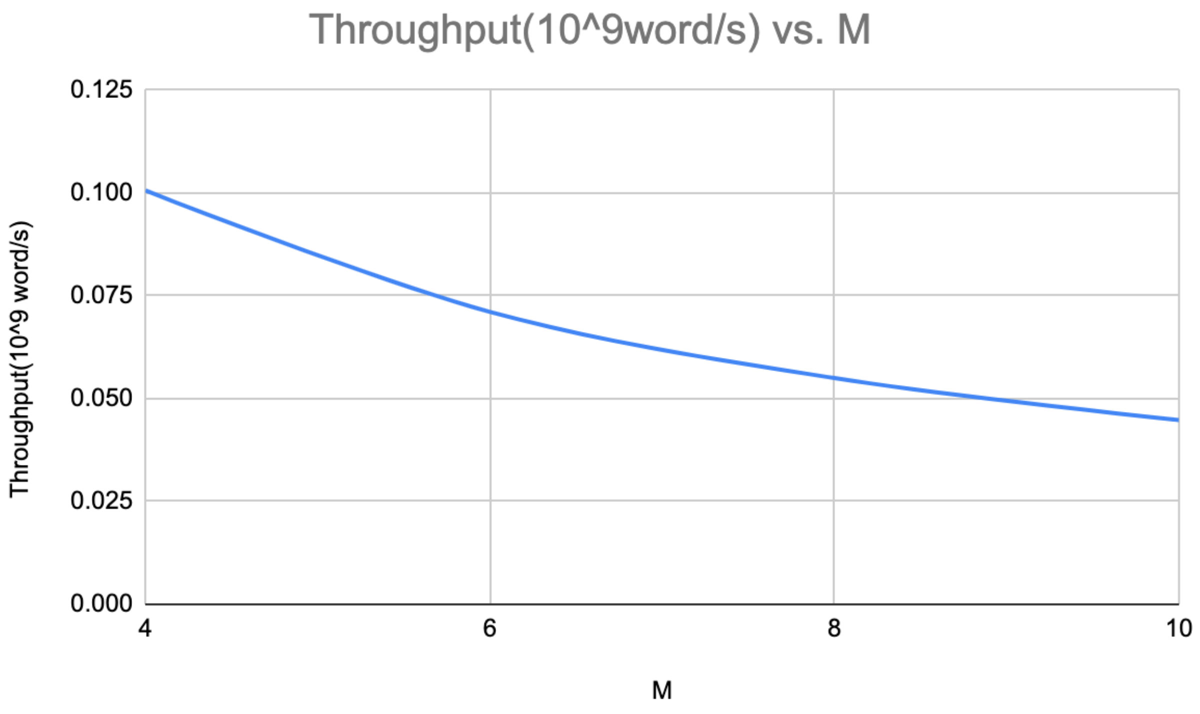

Figure 4.3: This is a plot showing the change in throughput as M increases.

The critical path shouldn’t change as M increases. As increasing the ROM size won’t affect the critical path. However, when M = 6 the critical path moved to the pipeline register to the accumulator output. The student made an assumption that the synthesis tool somehow optimized the datapath, such that the critical path changes for that instance. The clock cycles for one set of input finish processing are 53 cycles for M=4 , 75 cycles for M= 6, 97 cycles for M= 8 and 119 cycles for M =10. This can be predicted once the first one is known. The reason is that each time 2 more rows are being processed and it takes 11 cycles for one row being processed for our particular design. Furthermore, this also indicates that the throughput will decrease because the input is fixed but the ROM increases, thus there is a negative trend for the above throughput vs M graph. Increasing in M will cause the area and power to slightly increase due to the ROM size increases.

5.

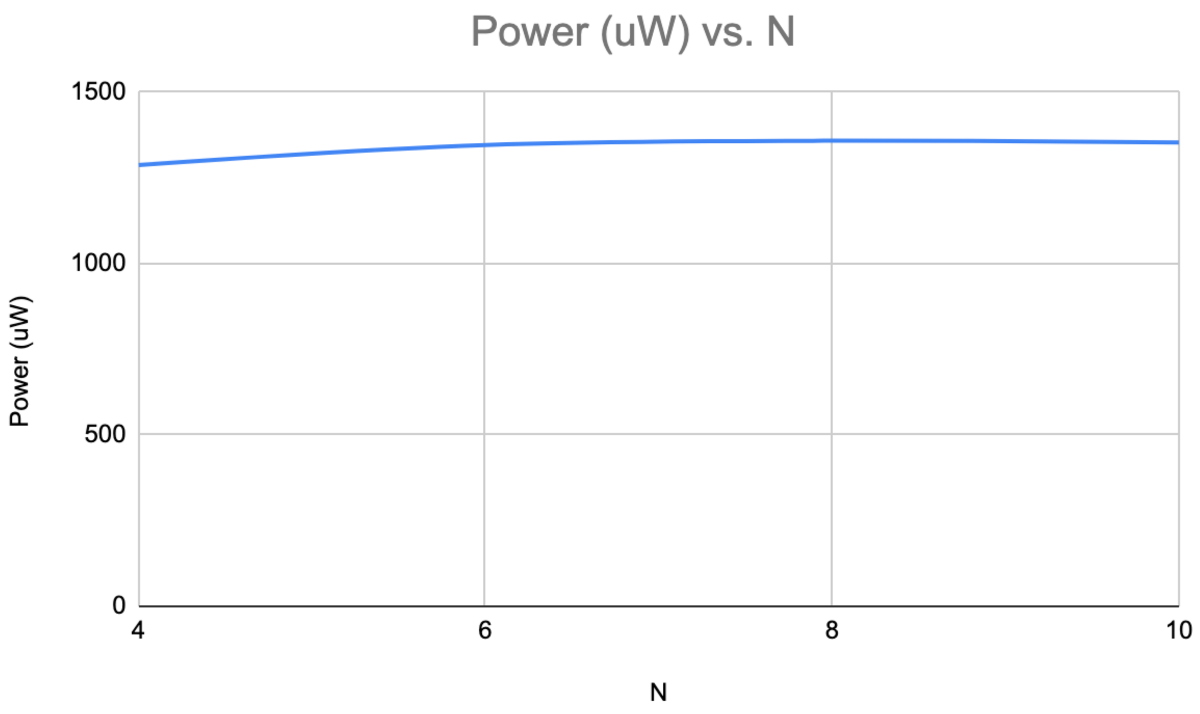

Figure 5.1: This is a plot showing the change in power as N changes.

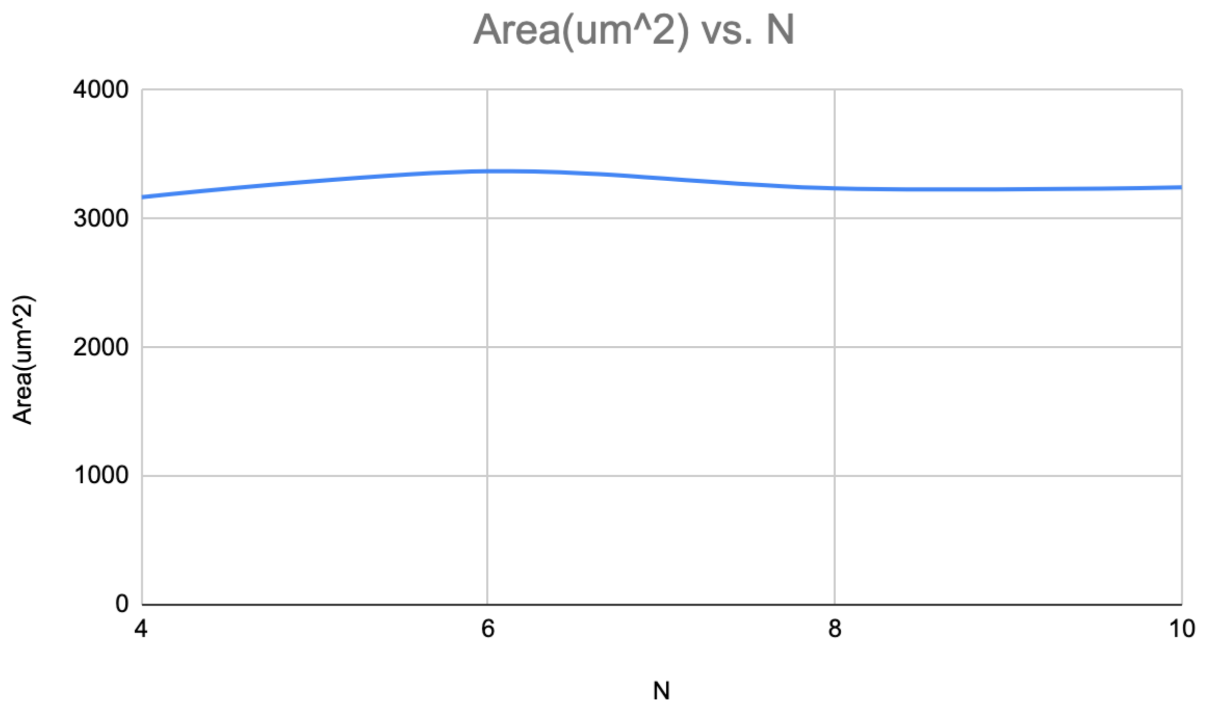

Figure 5.2: This is a plot showing the change in area as N increases.

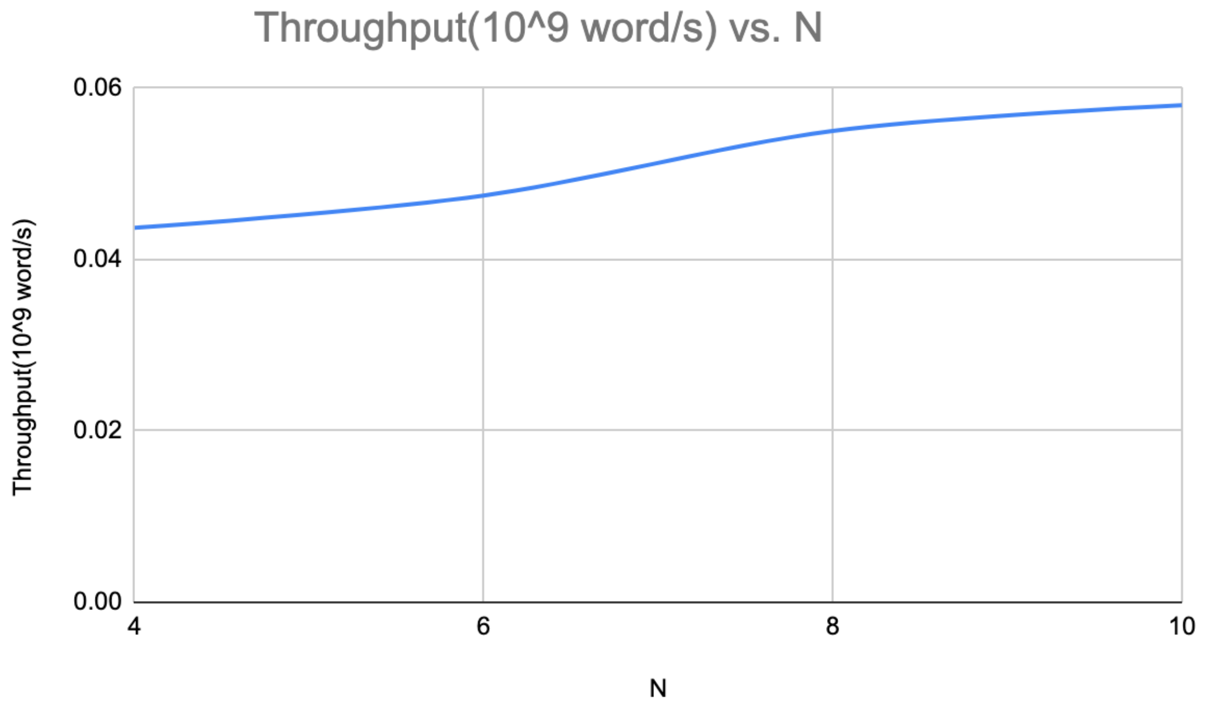

Figure 5.3: This is a plot showing the change in throughput as N increases.

5. The critical path does not change as N increases, it’s always going from the ROM register output to the pipeline register output. The reason is that increasing the vector size will not affect the critical path, thus no change.

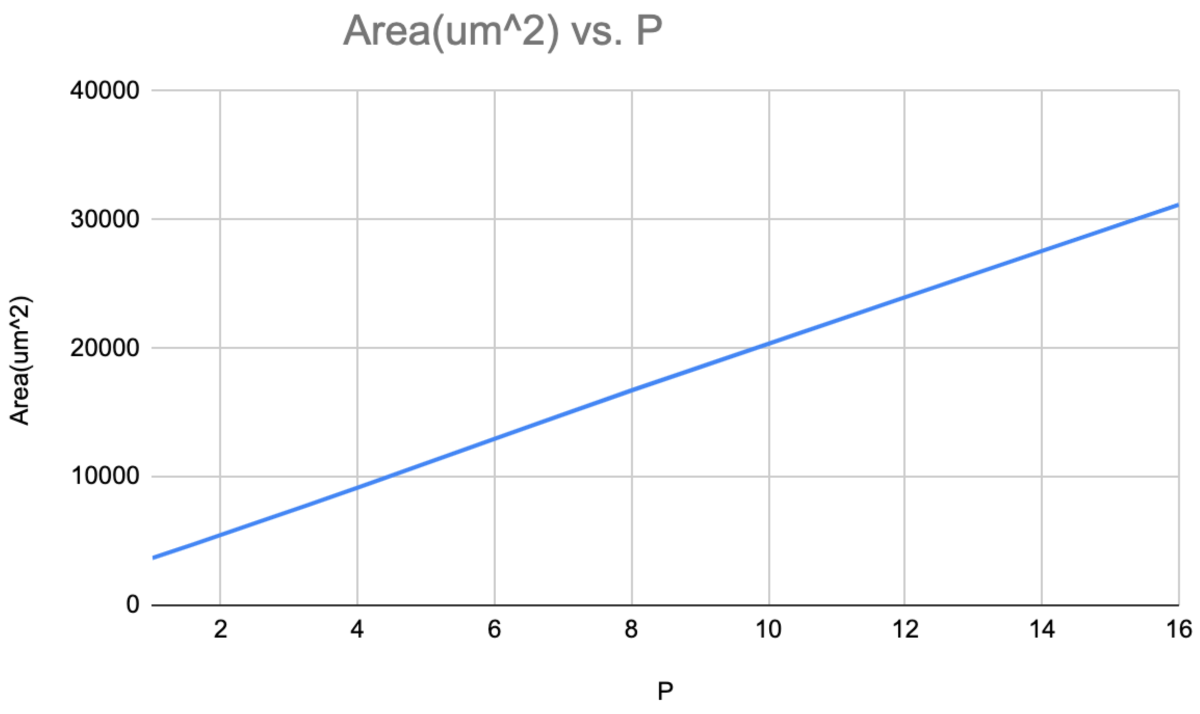

Figure 6.1: This is a plot showing the change in area as P changes.

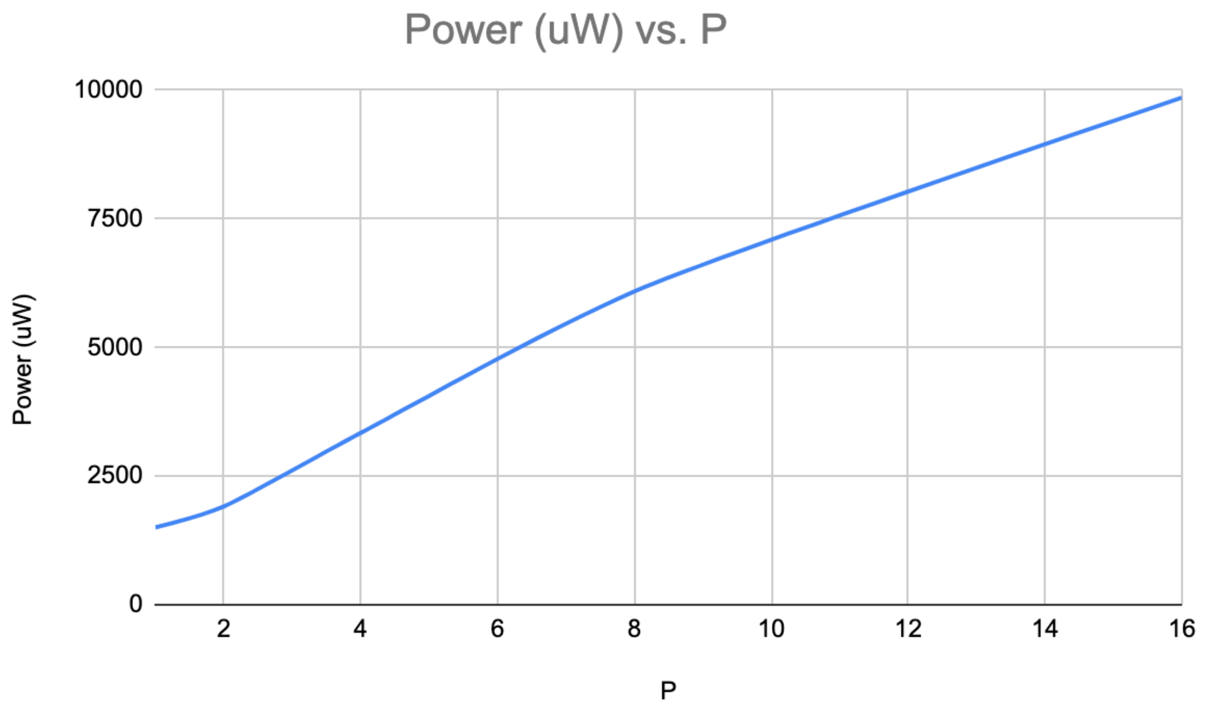

Figure 6.2: This is a plot showing the change in power in uW as P increases.

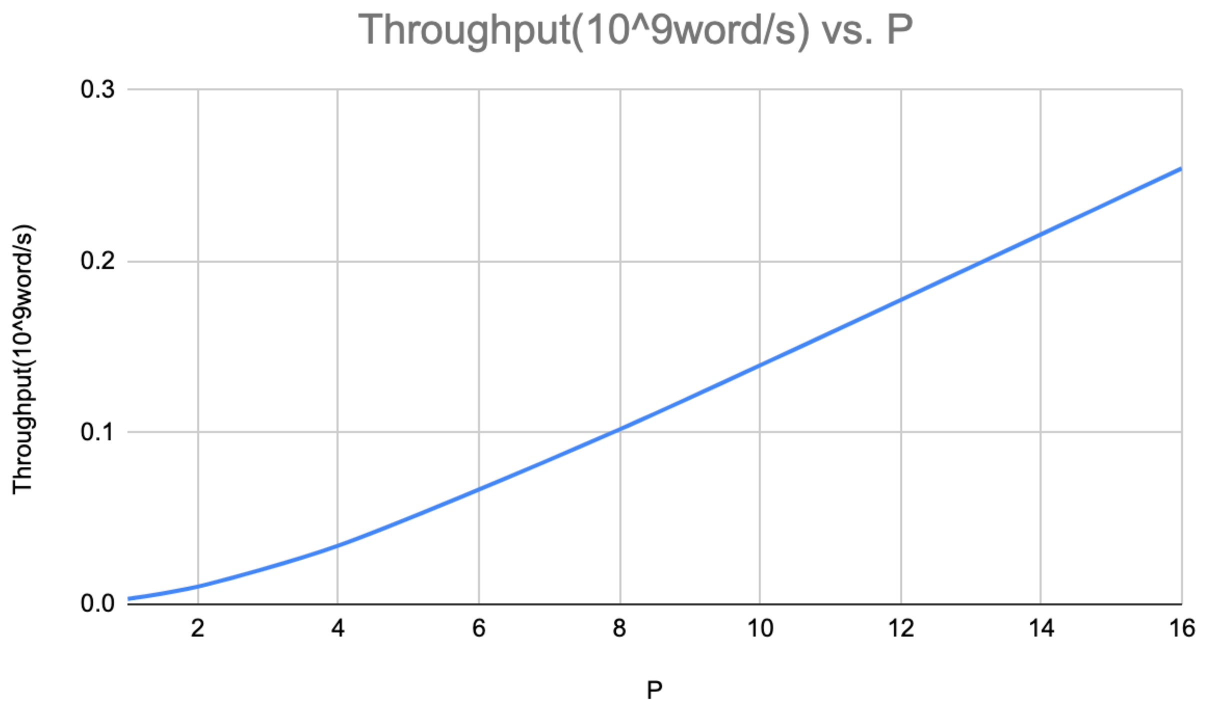

Figure 6.3: This is a plot showing the change in throughput as P increases.

6.
From both understanding of the circuit and the reading of the synthesis report, as the parallelism increases, there are more MAC units, which means the area will definitely increase. On the other hand, the power will also increase since power is directly related to the clock frequency. Pipeline increases the frequency which also increases the power. Lastly, the increase in throughput is a lot in comparison to the increase in area. The throughput increases with a ratio of 10^9 words.

7.
When P equals M, you cannot have any more MAC modules that are running in parallel to each other. Therefore the MAC module would need to be moduled to have multipliers that are running in parallel to each other if we were to further increase parallelism. The increase in multipliers also means that there would need to be an increase in memory modules to have an effect on computation speed. The adder or accumulator would also need to have more ports, as many as there are an increase in multipliers. Going as exhaustive as possible with the parallel computation, you could have N multipliers for each parallel computation of M. Resulting in a row being computed in 1 cycle rather than N cycles. The current design for one set of input data, the layer takes (Mi/Pi)*(N+4+Pi) to finish computation and outputting the values, by change N to 1, the clock cycle required will become (Mi/Pi)*(1+4+Pi), which saves (Mi/Pi)*(N-1) clock cycle. However, keep in mind that to have N multipliers concurrently, it will also increase the area, power and clock period(because there will require nested adders to add all the single product).

8.
The approach the student had was to first have 3 nested for loops, where each layer keeps increasing the pipeline number for each layer by one. Once the script has three values of Pi, then it will determine whether or not, Mi is divisible, if not keep increasing the Pi. Otherwise, it will also check the sum of Pi is still in the range of Budget B. If previous condition is met, then it will compute the clock cycle for each layer to compute and output a set of input value(since output from last layer is overlap with input from current layer, the student just also include the output time in this calculation). After this, the student will compare the relationship between the 3 clock cycle for each layer and based on the observation of the simulation, the student was able to calculate the lowest clock cycle based on different conditions of the 3 layer clock cycle relation. If the lowest clock cycle is smaller than the previous lowest, then the clock cycle and Pi combination will be recorded. Finally, the smallest slowest clock cycle generated by Pi combination will win and that Pi combination gets returned. This approach is a good solution based on the student's understanding of how each layer interacts with each other and the observation students gained from different simulations. There are some ways the student realized he can evaluate the wellness of the optimization, first of all, the student calculates other possible Pi parameters and have their clock cycle display in the c++ code(was commented out). On the other hand, the student can also brute forcely simulate all possible Pi combinations and validate with simulation waveform. Lastly, the student can also compare with friends about the optimization cycle to get a rough idea about his/her optimization result. Lastly, the student found that there could still be some improvement for his optimization. For the current approach the student has right now, once the optimization find the Pi values, it will not utilize the remaining

budget. Indeed, if the optimization can further use those budgets, then some relations between layers should further boost/reduce more clock cycles.

9.

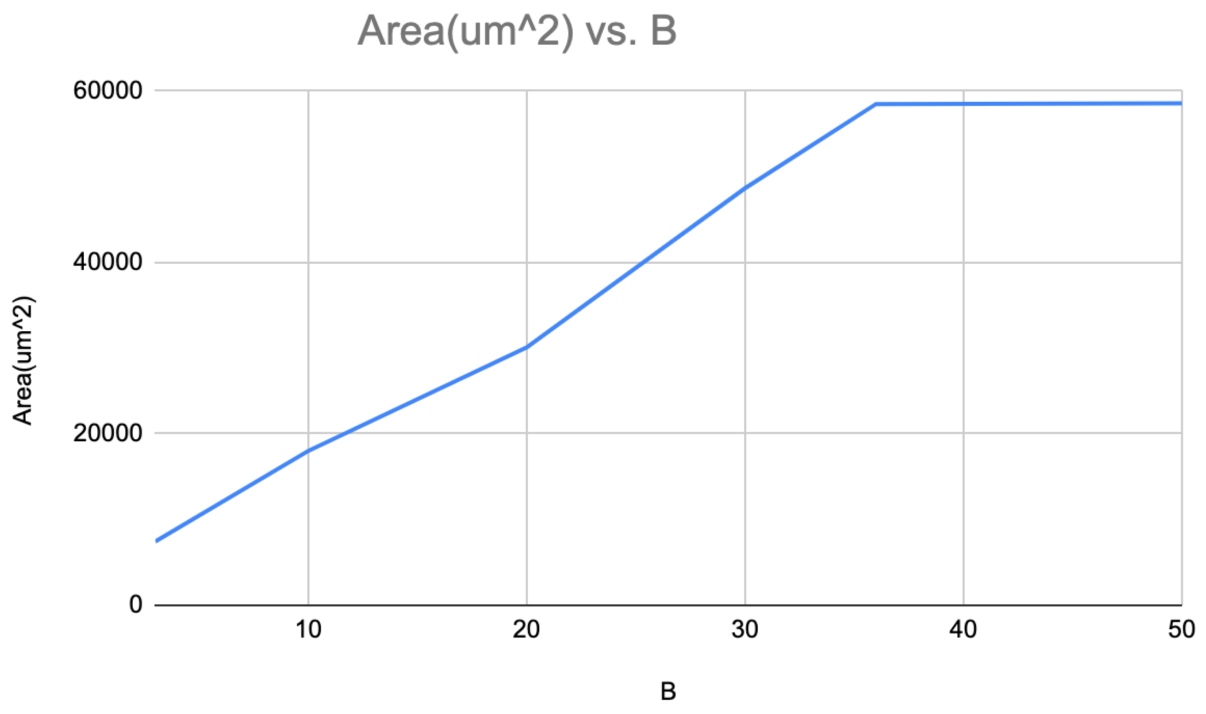

Figure 9.1: This is a plot showing the change in area as B changes.

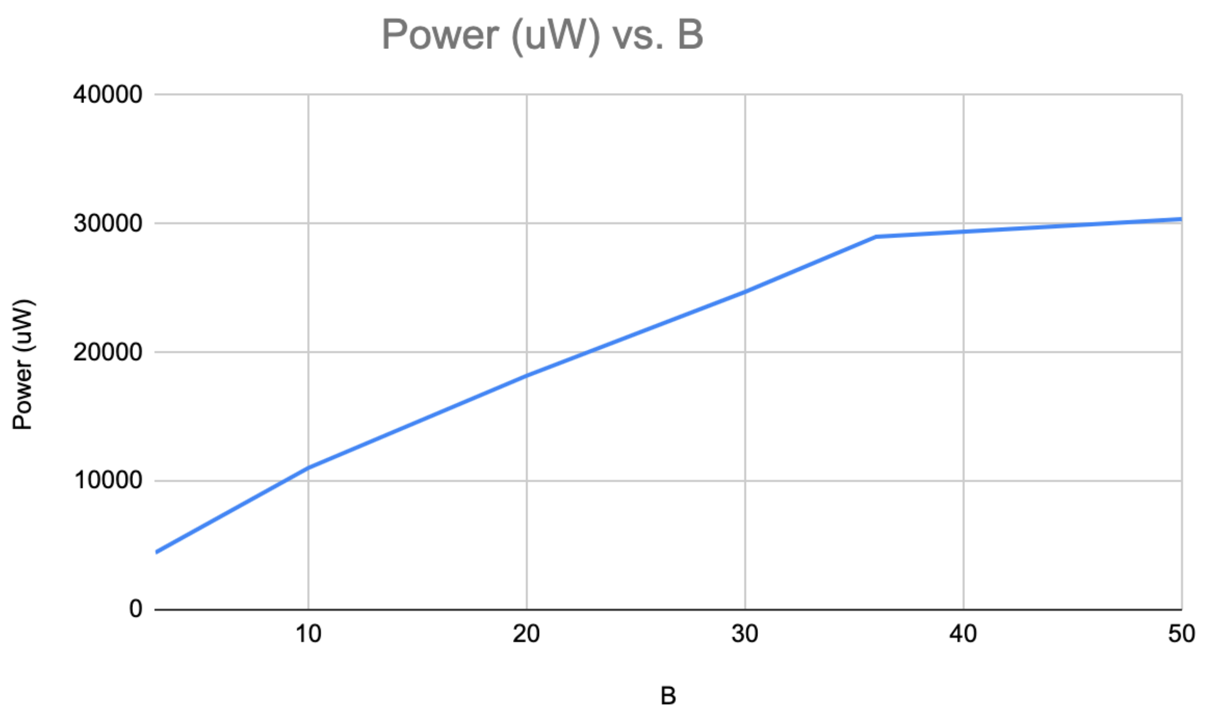

Figure 9.2: This is a plot showing the change in power as B changes.

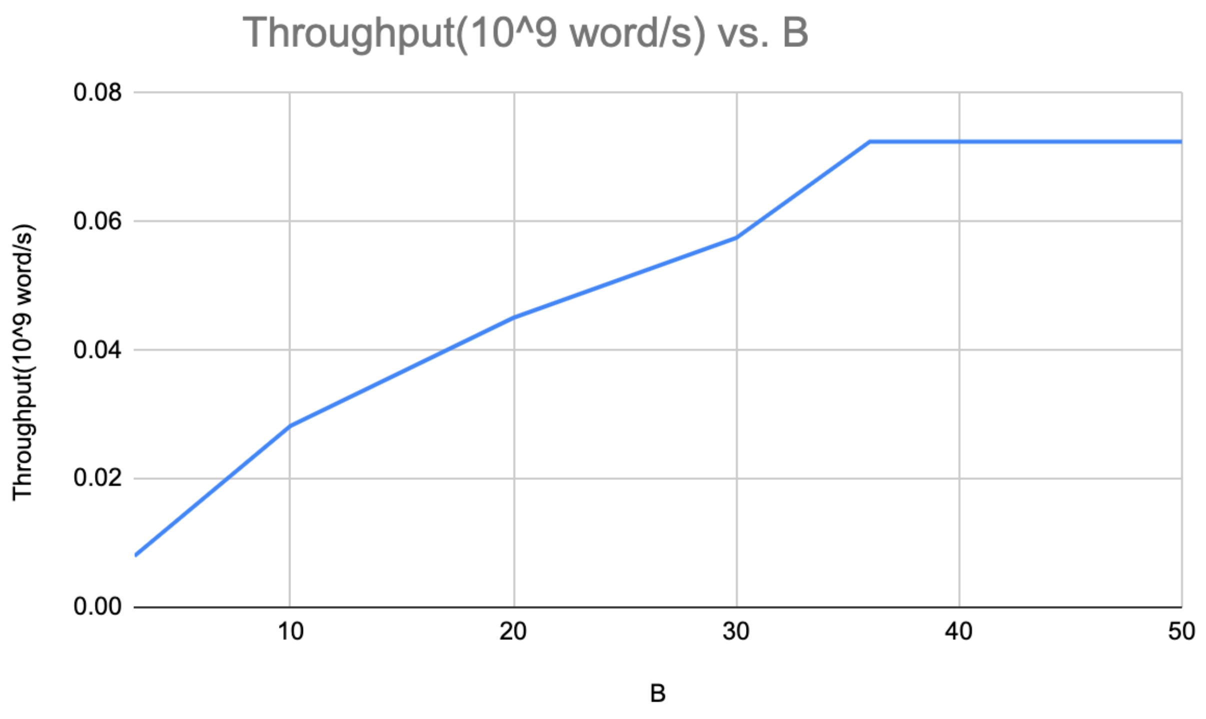

Figure 9.3: This is a plot showing the change in throughput as B changes.

9.Discussion: as seen in top figures, the throughput, area and power increases as the number of the budget increases. However, this trend somehow slows down after around B =30. This is because although the budget still increases up to 50, the optimization only chose to use 32 of them, because the optimization “thoughts” there would be no improvement in throughput to keep increasing the pipeline in certain layers.

10. There will be two things needed to modify such that accepting arbitrary layers. First of all, there should be a new parameter called L that stores the user specified number of layers. And then based on that value, a for loop call gets layers function l times. On the other hand, this parameter will also be needed to pass to the get rom function. There should be an array of Mi such that the function can cut the corresponding rom to the appropriate layers. The Mi values can be either user specified or evenly divides the ROM.

11. Student Jinxing Yin mainly focuses on project design and coding
Student Brian Cheung mainly focuses on analysis on the data